SOFTWARE DESIGN DOCUMENT

(HLD & Database)

for

Shopify Application

Arushi kumari , SACon
Infosys Limited
July 30 , 2024

1. INTRODUCTION ……1
1.1. PURPOSE……1
1.2. SCOPE……..1
1.3. OVERVIEW OF CONTENTS OF DOCUMENT……...2
2. ARCHITECTURAL DESIGN……..2
2.1 CLIENT / SERVER SOFTWARE LAYERS……2
3 HIGH LEVEL DESIGN…….3
3.2 APPLICATION OVERVIEW……...3
4 DATABASE DESIGN……4
5 Swagger UI……..5

Introduction

1.1. Purpose

The purpose of this document is to provide a detailed high-level design for the **Shopify Application**. This system tracks customer transactions and calculates reward points based on predefined rules. The application exposes RESTful APIs for managing customers, transactions, and reward points. Swagger UI is integrated to provide interactive API documentation.

This document aims to:

- Define the architecture and key components of the system.
- Describe the data model and database schema.
- Detail the API endpoints and their interactions.
- Outline the security configuration.
- Explain the data initialization process.
- Provide guidance for implementation and testing.

1.2. Scope

This document covers the design and implementation of the following aspects of the Reward Points System:

- **Customer Management**: Endpoints and functionality for customer registration, login, and logout.
- **Transaction Management**: Endpoints and functionality for managing customer transactions, including retrieval, addition, update, and deletion.
- **Reward Points Calculation**: Methods for calculating reward points based on transaction data, including total points per customer and monthly breakdown.
- **API Documentation**: Integration of Swagger UI for interactive API documentation and testing.
- **Security Configuration**: Details on how the application secures endpoints and integrates with Spring Security.
- **Database Design**: Schema and relationships for storing customer and transaction data.
- **Data Initialization**: Use of `CommandLineRunner` for loading dummy data into the database on application startup.

The scope does not include:

- Detailed implementation of third-party services or integrations.

- Comprehensive security testing beyond basic configurations.
- Performance optimizations beyond initial design considerations.

1.3. Overview of Contents of Document

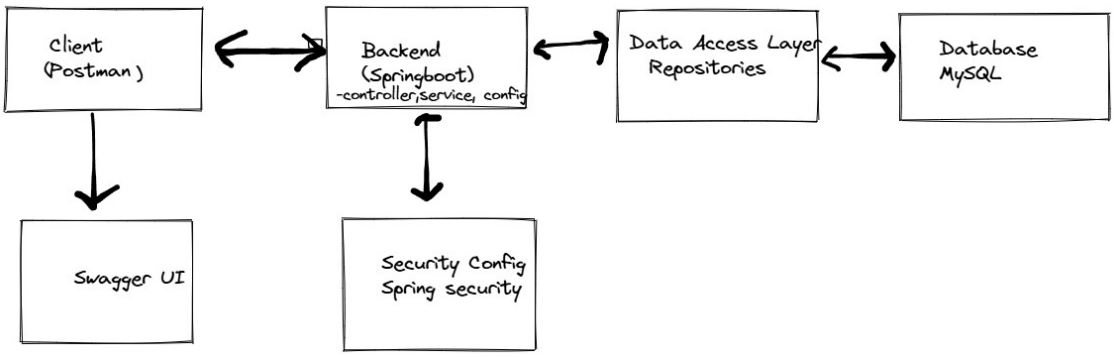
1. **System Overview**
    - Description of the system and its components.
    - Overview of interactions between components.
2. **Architecture**
    - High-level architectural diagram.
    - Description of major components and their interactions.

ARCHITECTURAL DESIGN

The system is composed of the following major components:

1. **Client**
    - **Postman**: Provides an interface for users to interact with the spring boot application. The client application communicates with the backend via RESTful APIs.
2. **Backend Application**
    - **Spring Boot Application**: Hosts the business logic and RESTful APIs. It handles requests from the client, processes them according to the business rules, and interacts with the database.
3. **Database**
    - **MySQL Database**: Stores customer and transaction data. The database schema is designed to efficiently handle queries related to transactions and reward points.
4. **Swagger UI**
    - **API Documentation**: Provides an interactive interface for exploring and testing the API endpoints. It helps in understanding the available endpoints and their usage.

High Level Design



Client

Purpose: To interact with the backend system, allowing users to perform operations like registering, logging in, and viewing their reward points.

Technology: web browser , Postman

Spring Boot Application:

Purpose: To implement the business logic, handle API requests, and interact with the database.

Components:

Controllers: Define RESTful endpoints for managing customers, transactions, and reward points.

Services: Contain business logic for calculating reward points and managing transactions.

Repositories: Interface with the database for CRUD operations.

Configuration: Includes security configurations and Swagger UI setup.

API Endpoints:

Customer Management:

- `POST /customers/register`: Register a new customer.
- `POST /customers/login`: Log in a customer.

- `POST /customers/logout`: Log out a customer.

Transaction Management:

- `GET /transactions/{customerId}`: Retrieve transactions for a customer.
- `POST /transactions`: Add a new transaction.
- `PUT /transactions/{id}`: Update a transaction.
- `DELETE /transactions/{id}`: Delete a transaction.

Reward Points Calculation:

- `GET /rewards/customer/{id}`: Get total reward points for a customer.
- `GET /rewards/monthly`: Get reward points per month for all customers.
- `GET /rewards/total`: Get total reward points for all customers.

Database Design

MySQL Database:

- **Purpose**: To store and manage customer and transaction data.
- **Schema**:

Customer Table: Stores customer details.

Fields:

- `id`: Unique identifier for the customer (Primary Key).
- `name`: Name of the customer.
- `email`: Email address of the customer (Unique).
- `password`: Password for customer authentication.
- `loggedIn`: Status indicating if the customer is currently logged in.

CustomerTransaction Table: Records transaction details.

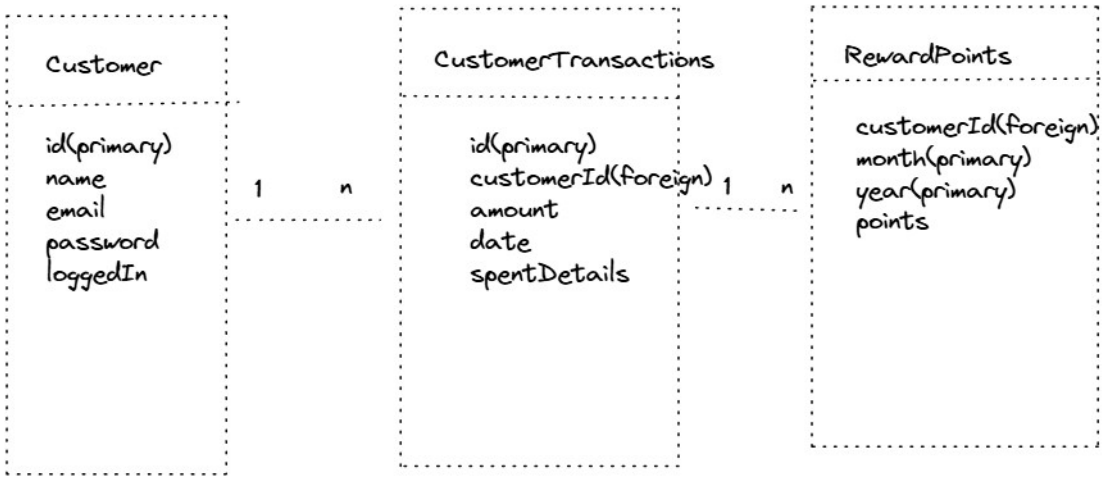
Fields:

- `customerId`: Identifier for the customer (Foreign Key).
- `month`: Month for which the points are calculated.
- `year`: Year for which the points are calculated.
- `points`: Total reward points for the customer in the given month and year.

RewardPoints Table: Stores calculated reward points.

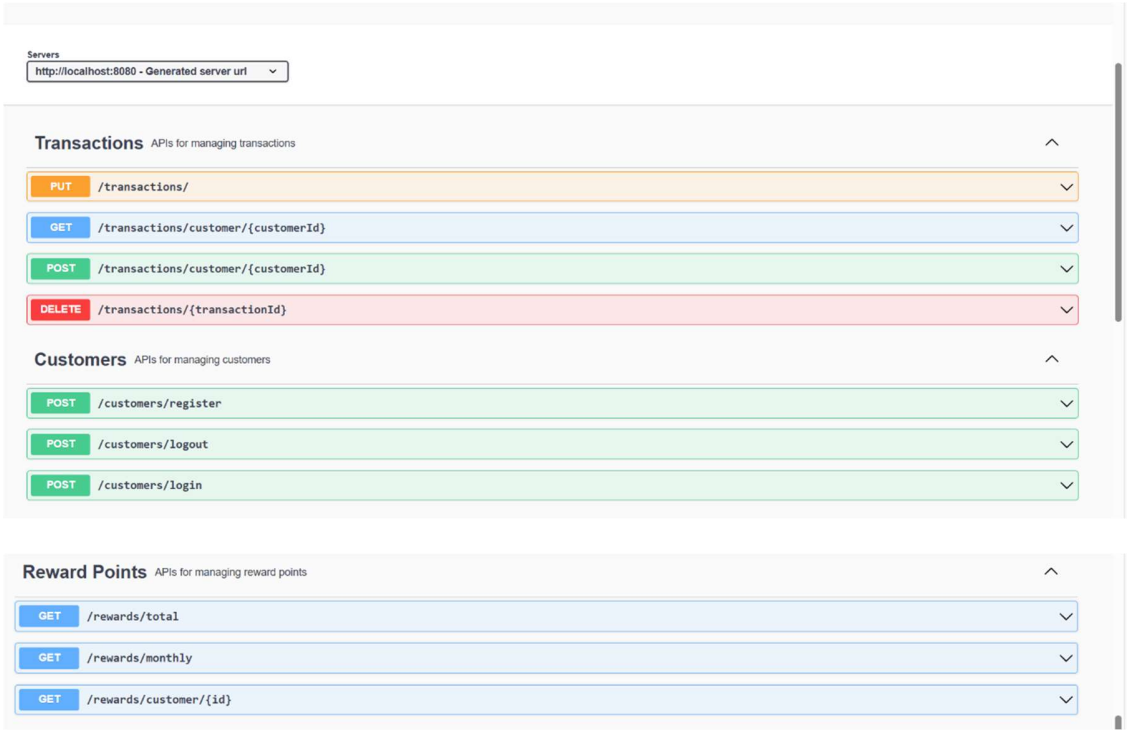
Fields:

- `customerId`: Identifier for the customer (Foreign Key).
- `month`: Month for which the points are calculated.
- `year`: Year for which the points are calculated.
- `points`: Total reward points for the customer in the given month and year.



Swagger UI

- **Purpose**: To provide interactive API documentation and testing.
- **Configuration**: Integrated via SpringDoc OpenAPI to automatically generate and display API documentation.


Servers
http://localhost:8080 - Generated server url
Transactions APIs for managing transactions
PUT /transactions/
GET /transactions/customer/{customerId}
POST /transactions/customer/{customerId}
DELETE /transactions/{transactionId}
Customers APIs for managing customers
POST /customers/register
POST /customers/logout
POST /customers/login

Reward Points APIs for managing reward points
GET /rewards/total
GET /rewards/monthly
GET /rewards/customer/{id}